Package ‘varycoef’

January 13, 2021

Type Package
Title Modeling Spatially Varying Coefficients
Version 0.3.0
Description Implements a maximum likelihood estimation (MLE) method for estimation and prediction of Gaussian process-based spatially varying coefficient (SVC) models (Dambon et al. (2021a) <doi:10.1016/j.spasta.2020.100470>). Covariance tapering (Furrer et al. (2006) <doi:10.1198/106186006X132178>) can be applied such that the method scales to large data. Further, it implements a joint variable selection of the fixed and random effects (Dambon et al. (2021b) <arXiv:2101.01932>).

License GPL-2
URL https://github.com/jakobdambon/varycoef
BugReports https://github.com/jakobdambon/varycoef/issues
Depends R (>= 3.5.0), spam
Imports DiceKriging, glmnet, lhs, mlr, mlrMBO, RandomFields, optimParallel (>= 0.8-1), ParamHelpers, pbapply, smoof, sp
Suggests gstat, knitr, microbenchmark, parallel, rmarkdown, R.rsp, spData
VignetteBuilder knitr, R.rsp
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
NeedsCompilation no
Author Jakob A. Dambon [aut, cre] (<https://orcid.org/0000-0001-5855-2017>), Fabio Sigrist [ctb] (<https://orcid.org/0000-0002-3994-2244>), Reinhard Furrer [ctb] (<https://orcid.org/0000-0002-6319-2332>)
Maintainer Jakob A. Dambon <jakob.dambon@math.uzh.ch>
Repository CRAN
Date/Publication 2021-01-13 11:40:02 UTC
R topics documented:

coef.SVC_mle .................................................. 2
cov_par ....................................................... 3
d.Lq ........................................................... 3
d.SCAD ......................................................... 4
fitted.SVC_mle .................................................. 5
fullSVC_regrid .................................................. 6
house ............................................................ 7
IC.SVC_mle ..................................................... 8
logLik.SVC_mle .................................................. 9
Lq ............................................................... 9
nlocs ............................................................ 10
nobs.SVC_mle .................................................... 11
plot.SVC_mle ................................................... 11
predict.SVC_mle ............................................... 13
print.summary.SVC_mle ....................................... 15
print.SVC_mle .................................................. 16
residuals.SVC_mle ............................................... 17
SCAD .......................................................... 17
summary.SVC_mle .............................................. 18
SVC_mle ........................................................ 19
SVC_mle_control ............................................... 22
SVC_selection .................................................. 25
SVC_selection_control ....................................... 26
varycoef ....................................................... 28

Index .......................................................... 30

---

c coef.SVC_mle Extact Mean Effects

Description

Method to extract the mean effects from an SVC_mle object.

Usage

## S3 method for class 'SVC_mle'
coef(object, ...)

Arguments

  object SVC_mle object
  ... further arguments

Value

named vector with mean effects, i.e. $\mu$ from SVC_mle
**cov_par**  

*Author(s)*  
Jakob Dambon

---

**cov_par**  

*Extract Covariance Parameters Function to extract the covariance parameters from an SVC_mle object.*

**Description**  

Extract Covariance Parameters  
Function to extract the covariance parameters from an SVC_mle object.

**Usage**  

cov_par(object, ...)

**Arguments**  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>SVC_mle object</td>
</tr>
<tr>
<td>...</td>
<td>further arguments</td>
</tr>
</tbody>
</table>

**Value**  

vector with covariance parameters and attributes what kind

- "GRF", character, describing the GRF used, see SVC_mle_control.
- "tapering", either NULL if no tapering is applied of the taper range.

**Author(s)**  
Jakob Dambon

---

**d.Lq**  

*Derivative of $L^q$ Norm Penalty*

**Description**  

derivative of $L_q$, which is not differentiable at $x = 0$.

**Usage**  

d.Lq(x, lambda = 1, q = 1, d.side = "both")
Arguments

\( x \) numeric.
\( \lambda \text{ lambda} \) non-negative scalar, shrinkage parameter.
\( q \) non-negative scalar, norm parameter.
\( \text{d.side} \) side of derivative at origin. Default value is "both", returning NA for \( x == 0 \). If set to "RHS", then returns RHS derivative, i.e., \( \lambda \), and \(-\lambda\) with "LHS".

Value

derivative of \( L^q(x) \), i.e.,
\[
L^q(x) = q \lambda |x|^{q-1}
\]
, for \( x == 0 \) return value is NA.

Author(s)

Jakob Dambon

Examples

d.Lq(-5:5)
d.Lq(-2:2, d.side = "LHS")curve(d.Lq, from = -5, to = 5)

---

d.SCAD

Derivative of Smoothly Clipped Absolute Deviation Penalty

Description

derivative of \( \text{SCAD} \), which is not differentiable at \( x == 0 \).

Usage

d.SCAD(x, lambda = 1, a = 3.7, d.side = "both")

Arguments

\( x \) numeric.
\( \lambda \text{ lambda} \) non-negative scalar, shrinkage parameter.
\( a \) scalar larger than 2. Fan & Li (2001) suggest \( a = 3.7 \).
\( \text{d.side} \) side of derivative at origin. Default value is "both", returning NA for \( x == 0 \). If set to "RHS", then returns RHS derivative, i.e., \( \lambda \), and \(-\lambda\) with "LHS".

Value

derivative of \( \text{SCAD}(x) \), for \( x == 0 \) return value is NA.
fitted.SVC_mle

Author(s)
Jakob Dambon

Examples

d.SCAD(-5:5)
d.SCAD(-2:2, d.side = "LHS")
curve(d.SCAD, from = -5, to = 5)

Description
Method to extract the fitted values from an SVC_mle object. This is only possible if save.fitted was set to TRUE in the control of the function call

Usage

## S3 method for class 'SVC_mle'
fitted(object, ...)

Arguments

object SVC_mle object
...
... further arguments

Value
Data frame, fitted values to given data, i.e., the SVC as well as the response and their locations

Author(s)
Jakob Dambon
fullSVC_reggrid

Sample Function for SVCs

Description

Samples SVCs on a regular quadratic (Cartesian) grid. The SVCs have all mean 0 and an exponential covariance function is used.

Usage

fullSVC_reggrid(m, p, cov_pars, nugget, seed = 123, given.locs = NULL)

Arguments

- **m** (numeric(1))
  Number of observations in one dimension, i.e., the square root number of total number of observation locations \( n = m^2 \).

- **p** (numeric(1))
  Number of SVCs.

- **cov_pars** (data.frame(p,2))
  Contains the covariance parameters of SVCs. The two columns must have the names "var" and "scale". These covariance parameters are then used for sampling the respective SVCs.

- **nugget** (numeric(1))
  Variance of the nugget / error term.

- **seed** (numeric(1))
  Seed set within the function for sampling.

- **given.locs** (NULL or data.frame(n,2))
  If NULL, the observations locations are sampled from a regular grid. Otherwise, the data frame contains the observation locations. The data frame must have two columns of name "x" and "y". The number of observations is then the number of rows \( n \).

Value

SpatialPointsDataFrame
(see SpatialPointsDataFrame-class) of the sampled SVC including the nugget.

Examples

```r
# number of SVC
p <- 3
# sqrt of total number of observations
m <- 20
# covariance parameters
(pars <- data.frame(var = c(0.1, 0.2, 0.3),
```

```r
```
Description

A dataset containing the prices and other attributes of 25,357 houses in Lucas County, Ohio. The selling dates span years 1993 to 1998. Data taken from house (spData package) and slightly modified to a data.frame.

Usage

house

Format

A data frame with 25357 rows and 25 variables:

- **price** (integer) selling price, in US dollars
- **yrbuilt** (integer) year the house was built
- **stories** (factor) levels are "one", "bilevel", "multilvl", "one+half", "two", "two+half", "three"
- **TLA** (integer) total living area, in square feet.
- **wall** (factor) levels are "stucdrvt", "ccbtile", "metlvnyl", "brick", "stone", "wood", "partbrk"
- **beds, baths, halfbaths** (integer) number of corresponding rooms / facilities.
- **frontage, depth** dimensions of the lot. Unit is feet.
- **garage** (factor) levels are "no garage", "basement", "attached", "detached", "carport"
- **garagesqft** (integer) garage area, in square feet. If garage == "no garage", then garagesqft == 0.
- **rooms** (integer) number of rooms
- **lotsize** (integer) area of lot, in square feet
- **sdate** (Date) selling date, in format yyyy-mm-dd
- **avalue** (int) appraised value
long, lat (numeric) location of houses. Longitude and Latitude are given in CRS(+init=epsg:2834), the Ohio North State Plane. Units are meters.

Source


IC.SVC_mle

Conditioned Akaike’s and Bayesian Information Criteria

Description

Methods to calculate information criteria for SVC_mle objects. Currently, two are supported: the conditioned Akaike’s Information Criteria \( cAIC = -2 \times \log - \text{likelihood} + 2 \times (edof + df) \) and the Bayesian Information Criteria \( BIC = -2 \times \log - \text{likelihood} + \log(n) \times npar \). Note that the Akaike’s Information Criteria is of the corrected form, that is: \( edof \) is the effective degrees of freedom which is derived as the trace of the hat matrices and \( df \) is the degree of freedoms with respect to mean parameters.

Usage

## S3 method for class 'SVC_mle'
BIC(object, ...)

## S3 method for class 'SVC_mle'
AIC(object, conditional = "BW", ...)

Arguments

object SVC_mle object
...
... further arguments
conditional string. If conditional = "BW", the conditional AIC is calculated.

Value

numeric, value of information criteria

Author(s)

Jakob Dambon
**logLik.SVC_mle**

*Extract the Likelihood*

**Description**

Method to extract the computed (penalized) log (profile) Likelihood from an *SVC_mle* object.

**Usage**

```r
## S3 method for class 'SVC_mle'
logLik(object, ...)  
```

**Arguments**

- `object` *SVC_mle* object
- `...` further arguments

**Value**

An object of class `logLik` with attributes

- "penalized", logical, if the likelihood (FALSE) or some penalized likelihood (TRUE) was optimized.
- "profileLik", logical, if the optimization was done using the profile likelihood (TRUE) or not.
- "nobs", integer of number of observations
- "df", integer of how many parameters were estimated. **Note**: This includes only the covariance parameters if the profile likelihood was used.

**Author(s)**

Jakob Dambon

---

**Lq**

*\(L^q\) Norm Penalty*

**Description**

Penalty function using the \(L^q\) norm, i.e., \(p_\lambda(x) = \lambda \|x\|^q\).

**Usage**

```r
Lq(x, lambda = 1, q = 1)
```
Arguments

- **x** numeric.
- **lambda** non-negative scalar, shrinkage parameter.
- **q** non-negative scalar, norm parameter.

Value

penalty for values of x.

Author(s)

Jakob Dambon

Examples

Lq(-5:5)
curve(Lq(x, q = 2), from = -5, to = 5)

**nlocs**

*Extract Number of Unique Locations Function to extract the number of unique locations in the data set used in an MLE of the SVC_mle object.*

Description

Extract Number of Unique Locations

Function to extract the number of unique locations in the data set used in an MLE of the SVC_mle object.

Usage

nlocs(object)

Arguments

- **object** SVC_mle object

Value

integer with the number of unique locations

Author(s)

Jakob Dambon
nobs.SVC_mle

**nobs.SVC_mle**

*Extract Number of Observations*

---

**Description**

Method to extract the number of observations used in MLE for an *SVC_mle* object.

**Usage**

```r
## S3 method for class 'SVC_mle'
nobs(object, ...)
```

**Arguments**

- `object` *SVC_mle* object
- `...` further arguments

**Value**

an integer of number of observations

**Author(s)**

Jakob Dambon

---

plot.SVC_mle

**plot.SVC_mle**

*Plotting Residuals of SVC_mle model*

---

**Description**

Method to plot the residuals from an *SVC_mle* object. For this, `save.fitted` has to be `TRUE` in `SVC_mle_control`.

**Usage**

```r
## S3 method for class 'SVC_mle'
plot(x, which = 1:3, legend.pos = "bottomright", ...)
```

**Arguments**

- `x` *SVC_mle* object
- `which` numeric, indicating which of the 3 plots should be plotted
- `legend.pos` character describing the position of the legend in the spatial residual plot, see `legend`
- `...` further arguments
Value

a maximum 3 plots

- Tukey-Anscombe plot, i.e. residuals vs. fitted
- QQ-plot
- spatial residuals

Author(s)

Jakob Dambon

See Also

legend SVC_mle

Examples

```r
#' #
#  ---- toy example ----
## sample data
# setting seed for reproducibility
set.seed(123)
m <- 7
# number of observations
n <- m*m
# number of SVC
p <- 3
# sample data
y <- rnorm(n)
X <- matrix(rnorm(n*p), ncol = p)
# locations on a regular m-by-m-grid
locs <- expand.grid(seq(0, 1, length.out = m),
                      seq(0, 1, length.out = m))

## preparing for maximum likelihood estimation (MLE)
# controls specific to MLE
control <- SVC_mle_control(
    # initial values of optimization
    init = rep(0.1, 2*p+1),
    # using profile likelihood
    profileLik = TRUE)

# controls specific to optimization procedure, see help(optim)
opt.control <- list(
    # number of iterations (set to one for demonstration sake)
    maxit = 1,
    # tracing information
    trace = 6)

## starting MLE
```
predict.SVC_mle

```r
fit <- SVC_mle(y = y, X = X, locs = locs,
          control = control,
          optim.control = opt.control)

## output: convergence code equal to 1, since maxit was only 1
summary(fit)

## plot residuals
# only QQ-plot
plot(fit, which = 2)

# all three plots next to each other
oldpar <- par(mfrow = c(1, 3))
plot(fit)
par(oldpar)
```

---

**predict.SVC_mle**  
*Prediction of SVCs (and response variable)*

**Description**

Prediction of SVCs (and response variable)

**Usage**

```r
## S3 method for class 'SVC_mle'
predict(
  object,
  newlocs = NULL,
  newX = NULL,
  newW = NULL,
  compute.y.var = FALSE,
  ...
)
```

**Arguments**

- **object**  
  (SVC_mle)  
  Model obtained from **SVC_mle** function call.

- **newlocs**  
  (NULL or matrix(n.new,2))  
  If NULL, then function uses observed locations of model to estimate SVCs. Otherwise, these are the new locations the SVCs are predicted for.

- **newX**  
  (NULL or matrix(n.new,q))  
  If provided (together with newW), the function also returns the predicted response variable.
predict.SVC_mle

newW (NULL or matrix(n.new,p))
If provided (together with newX), the function also returns the predicted response variable.

compute.y.var (logical(1))
If TRUE and the response is being estimated, the predictive variance of each estimate will be computed.

... further arguments

Value

The function returns a data frame of n.new rows and with columns

- SVC_1,...,SVC_p: the predicted SVC at locations newlocs.
- y.pred, if newX and newW are provided
- y.var, if newX and newW are provided and compute.y.var is set to TRUE.
- loc_x,loc_y, the locations of the predictions

Author(s)

Jakob Dambon

References


See Also

SVC_mle

Examples

```r
## ---- toy example ----
## sample data
# setting seed for reproducibility
set.seed(123)
m <- 7  # number of observations
n <- m^2
# number of SVC
p <- 3
# sample data
y <- rnorm(n)
X <- matrix(rnorm(n*p), ncol = p)
# locations on a regular m-by-m-grid
locs <- expand.grid(seq(0, 1, length.out = m),
  seq(0, 1, length.out = m))

## preparing for maximum likelihood estimation (MLE)
```
# controls specific to MLE
control <- SVC_mle_control(
    # initial values of optimization
    init = rep(0.1, 2*p+1),
    # lower bound
    lower = rep(1e-6, 2*p+1),
    # using profile likelihood
    profileLik = TRUE
)

# controls specific to optimization procedure, see help(optim)
opt.control <- list(
    # number of iterations (set to one for demonstration sake)
    maxit = 1,
    # tracing information
    trace = 6
)

## starting MLE
fit <- SVC_mle(y = y, X = X, locs = locs,
    control = control,
    optim.control = opt.control)

## output: convergence code equal to 1, since maxit was only 1
summary(fit)

## prediction
# new location
newlocs <- matrix(0.5, ncol = 2, nrow = 2)

# new data
X.new <- matrix(rnorm(2*p), ncol = p)

# predicting SVCs
predict(fit, newlocs = newlocs)

# predicting SVCs and calculating response
predict(fit, newlocs = newlocs,
    newX = X.new, newW = X.new)

# predicting SVCs, calculating response and predictive variance
predict(fit, newlocs = newlocs,
    newX = X.new, newW = X.new,
    compute.y.var = TRUE)

print.summary.SVC_mle  
Printing Method for summary.SVC_mle

**Description**
Printing Method for summary.SVC_mle
print.SVC_mle

Usage

## S3 method for class 'summary.SVC_mle'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

Arguments

x  summary.SVC_mle
digits  the number of significant digits to use when printing.
...  further arguments

Value

The printed output of the summary in the console.

See Also

summary.SVC_mle SVC_mle

Description

Method to print an SVC_mle object.

Usage

## S3 method for class 'SVC_mle'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

Arguments

x  SVC_mle object
digits  (numeric) Number of digits to be plotted.
...  further arguments

Author(s)

Jakob Dambon
residuals.SVC_mle

**Description**

Method to extract the residuals from an \texttt{SVC\_mle} object. This is only possible if \texttt{save.fitted} was set to \texttt{TRUE}.

**Usage**

```r
## S3 method for class 'SVC_mle'
residuals(object, ...)
```

**Arguments**

- `object` \texttt{SVC\_mle} object
- `...` further arguments

**Value**

\((\text{numeric}(n))\) Residuals of model

**Author(s)**

Jakob Dambon

---

**SCAD**

\textit{Smoothly Clipped Absolute Deviation Penalty}

**Description**


**Usage**

```r
SCAD(x, lambda = 1, a = 3.7)
```

**Arguments**

- `x` numeric.
- `lambda` non-negative scalar, shrinkage parameter.
- `a` scalar larger than 2. Fan & Li (2001) suggest \(a = 3.7\).

**Value**

penalty for values of \(x\).
Author(s)
Jakob Dambon

References

Examples

```r
SCAD(-5:5)
curve(SCAD, from = -5, to = 5)
```

---

### summary.SVC_mle

#### Summary Method for SVC_mle

**Description**
Method to construct a `summary.SVC_mle` object out of a `SVC_mle` object.

**Usage**

```r
## S3 method for class 'SVC_mle'
summary(object, ...)
```

**Arguments**

- `object` `SVC_mle` object
- `...` further arguments

**Value**
object of class `summary.SVC_mle` with summarized values of the MLE.

**Author(s)**
Jakob Dambon

**See Also**
`SVC_mle`
**Description**

Conducts a maximum likelihood estimation (MLE) for a Gaussian process-based SVC model as described in Dambon et al. (2021) doi: 10.1016/j.spasta.2020.100470. More specifically, the model is defined as:

\[
y(s) = X\mu + W\eta(s) + \epsilon(s)
\]

where:

- \( y \) is the response (vector of length \( n \))
- \( X \) is the data matrix for the fixed effects covariates. The dimensions are \( n \) times \( q \). This leads to \( q \) fixed effects.
- \( \mu \) is the vector containing the fixed effects
- \( W \) is the data matrix for the SVCs modeled by GPs. The dimensions are \( n \) times \( p \). This leads to \( p \) SVCs in the model.
- \( \eta \) are the SVCs represented by a GP.
- \( \epsilon \) is the nugget effect

The MLE is an numeric optimization that runs `optim` or (if parallelized) `optimParallel`.

**Usage**

SVC_mle(...)

```r
## Default S3 method:
SVC_mle(y, X, locs, W = NULL, control = NULL, optim.control = list(), ...)

## S3 method for class 'formula'
SVC_mle(
  formula, data, 
  RE_formula = NULL, locs, 
  control, optim.control = list(), ...
)
```
Arguments

... further arguments

y  (numeric(n))
Response vector.

X  (matrix(n,q))
Design matrix. Intercept has to be added manually.

locs  (matrix(n,d))
Locations in a $d$-dimensional space. May contain multiple observations at single location.

W  (NULL or matrix(n,p))
If NULL, the same matrix as provided in $X$ is used. This fits a full SVC model, i.e., each covariate effect is modeled with a mean and an SVC. In this case we have $p=q$. If optional matrix $W$ is provided, SVCs are only modeled for covariates within matrix $W$.

c control  (list)
Control parameters given by SVC_mle_control.

optim.control  (list)
Control arguments for optimization function, see Details in optim.

formula  Formula describing the fixed effects in SVC model. The response, i.e. LHS of the formula, is not allowed to have functions such as sqrt() or log().

data  data frame containing the observations

RE_formula  Formula describing the random effects in SVC model. Only RHS is considered. If NULL, the same RHS of argument formula for fixed effects is used.

Value

Object of class SVC_mle if control$extract_fun = FALSE, meaning that a MLE has been conducted. Otherwise, if control$extract_fun = TRUE, the function returns a list with two entries:

- obj_fun: the objective function used in the optimization
- args: the arguments to evaluate the objective function.

For further details, see description of SVC_mle_control.

Author(s)

Jakob Dambon

References


See Also

predict.SVC_mle
Examples

```r
## ---- toy example ----
## sample data
# setting seed for reproducibility
set.seed(123)
m <- 7
# number of observations
n <- m*m
# number of SVC
p <- 3
# sample data
y <- rnorm(n)
X <- matrix(rnorm(n*p), ncol = p)
# locations on a regular m-by-m-grid
locs <- expand.grid(seq(0, 1, length.out = m),
seq(0, 1, length.out = m))

## preparing for maximum likelihood estimation (MLE)
# controls specific to MLE
control <- SVC_mle_control(
  # initial values of optimization
  init = rep(0.1, 2*p+1),
  # lower bound
  lower = rep(1e-6, 2*p+1),
  # using profile likelihood
  profileLik = TRUE)

# controls specific to optimization procedure, see help(optim)
opt.control <- list(
  # number of iterations (set to one for demonstration sake)
  maxit = 1,
  # tracing information
  trace = 6
)

## starting MLE
fit <- SVC_mle(y = y, X = X, locs = locs,
control = control,
optim.control = opt.control)

class(fit)

## output: convergence code equal to 1, since maxit was only 1
summary(fit)

## extract the optimization arguments, including objective function
control$extract_fun <- TRUE
opt <- SVC_mle(y = y, X = X, locs = locs,
control = control)

# objective function and its arguments of optimization
class(opt$obj.fun)
```
class(opt$args)

# single evaluation with initial value
do.call(opt$obj_fun,
    c(list(x = control$init), opt$args))

## ---- real data example ----
require(sp)
## get data set
data("meuse", package = "sp")

# construct data matrix and response, scale locations
y <- log(meuse$cadmium)
X <- model.matrix(~1+dist+lime+elev, data = meuse)
locs <- as.matrix(meuse[, 1:2])/1000

## starting MLE
# the next call takes a couple of seconds
fit <- SVC_mle(y = y, X = X, locs = locs,
    # has 4 fixed effects, but only 3 random effects (SVC)
    # elev is missing in SVC
    W = X[, 1:3],
    control = SVC_mle_control(
        # initial values for 3 SVC
        # 7 = (3 * 2 covariance parameters + nugget)
        init = c(rep(c(0.4, 0.2), 3), 0.2),
        profileLik = TRUE
    ))

## summary and residual output
summary(fit)
plot(fit)

## predict
# new locations
newlocs <- expand.grid(
    x = seq(min(locs[, 1]), max(locs[, 1]), length.out = 30),
    y = seq(min(locs[, 2]), max(locs[, 2]), length.out = 30))
# predict SVC for new locations
SVC <- predict(fit, newlocs = as.matrix(newlocs))
# visualization
sp.SVC <- SVC
coordinates(sp.SVC) <- ~loc_x+loc_y
spplot(sp.SVC, colorkey = TRUE)
**Description**

Function to set up control parameters for `SVC_mle`. In the following, we assume the SVC model to have \( p \) GPs, which model the SVCs, and \( q \) fixed effects.

**Usage**

`SVC_mle_control(...)`

```r
## Default S3 method:
SVC_mle_control(
  cov.name = c("exp", "sph"),
  tapering = NULL,
  parallel = NULL,
  init = NULL,
  lower = NULL,
  upper = NULL,
  save.fitted = TRUE,
  profileLik = FALSE,
  mean.est = c("GLS", "OLS"),
  pc.prior = NULL,
  extract.fun = FALSE,
  hessian = FALSE,
  dist = list(method = "euclidean"),
  ...
)
## S3 method for class 'SVC_mle'
SVC_mle_control(object, ...)
```

**Arguments**

- `...` further parameters yet to be implemented
- `cov.name` (character(1)) Name of the covariance function defining the covariance matrix of the GRF. Currently, only "exp" for the exponential and "exp" for spherical covariance functions are supported.
- `tapering` (NULL or numeric(1)) If NULL, no tapering is applied. If a scalar is given, covariance tapering with this taper range is applied, for all Gaussian processes modelling the SVC.
- `parallel` (NULL or list) If NULL, no parallelization is applied. If cluster has been established, define arguments for parallelization with a list, see documentation of `optimParallel`.
- `init` (numeric(2p+1q+as.numeric(profileLik))) Initial values for optimization procedure. The vector consists of \( p \)-times (alternating) scale and variance, the nugget variance and (if `profileLik = TRUE`) \( q \) mean effects.
lower (NULL or numeric(2p+1+q+as.numeric(profileLik)))
Lower bound for init in optim. Default NULL sets the lower bounds to 1e-05 for range and nugget parameters, 0 for variance parameters and -Inf for mean parameters.

upper (NULL or numeric(2p+1+q+as.numeric(profileLik)))
Upper bound for init in optim. Default NULL sets the upper bounds for all parameters to Inf.

save.fitted (logical(1))
If TRUE, calculates the fitted values and residuals after MLE and stores them. This is necessary to call residuals and fitted methods afterwards.

profileLik (logical(1))
If TRUE, MLE is done over profile Likelihood of covariance parameters.

mean.est (character(1))
If profileLik = TRUE, the means have to be estimated separately for each step. "GLS" uses the generalized least square estimate while "OLS" uses the ordinary least squares estimate.

pc.prior (NULL or numeric(4))
If numeric vector is given, penalized complexity priors are applied. The order is $\rho_0, \alpha_\rho, \sigma_0, \alpha_\sigma$, to give some prior believe for the range and the standard deviation of GPs, such that $P(\rho < \rho_0) = \alpha_\rho, P(\sigma > \sigma_0) = \alpha_\sigma$. This regulates the optimization process. Currently, only supported for GPs with of Matérn class covariance functions. Based on the idea by Fulgstad et al. (2018) doi: 10.1080/01621459.2017.1415907.

extract_fun (logical(1))
If TRUE, the function call of SVC_mle stops before the MLE and gives back the objective function of the MLE as well as all used arguments. If FALSE, regular MLE is conducted.

hessian (logical(1))
If FALSE, Hessian matrix is computed, see optim.

dist (list)
List containing the arguments of nearestdist. This controls the method of how the distances and therefore dependency structures are calculated. The default gives Euclidean distances in a d-dimensional space. Further editable arguments are p, miles, R, see help file of nearestdist. The other arguments, i.e., x, y, delta, upper, are set and not to be altered. Without tapering, delta is set to 1e-99.

object (SVC_mle)
The function then extracts the control settings from the function call used to compute in the given SVC_mle object.

Details

The argument extract_fun is useful, when one wants to modify the objective function. Further, when trying to parallelize the optimization, it is useful to check whether a single evaluation of the objective function takes longer than 0.05 seconds to evaluate, cf. Gerber and Furrer (2019) doi: 10.32614/RJ2019030. Platform specific issues can be sorted out by the user by setting up their own optimization.
SVC_selection

Value
A list with which SVC_mle can be controlled.

Author(s)
Jakob Dambon

See Also
SVC_mle

Examples
control <- SVC_mle_control(init = rep(0.3, 10))
# or
ccontrol <- SVC_mle_control()
ccontrol$init <- rep(0.3, 10)

SVC_selection  SVC Model Selection

Description
This function implements the variable selection for Gaussian process-based SVC models using a penalized maximum likelihood estimation (PMLE, Dambon et al., 2021, <arXiv:2101.01932>). It jointly selects the fixed and random effects of GP-based SVC models.

Usage
SVC_selection(obj.fun, mle.par, control, ...)

Arguments

obj.fun  (SVC_obj_fun)
Function of class SVC_obj_fun. This is the output of SVC_mle with the SVC_mle_control parameter extract_fun set to TRUE. This objective function comprises of the whole SVC model on which the selection should be applied.

mle.par  (numeric(2*q+1))
Numeric vector with estimated covariance parameters of unpenalized MLE.

control  (list)
List of control parameters for variable selection. Output of SVC_selection_control.

...
Further arguments.

Value
Returns an object of class SVC_selection. It contains parameter estimates under PMLE and the optimization as well as choice of the shrinkage parameters.
**Author(s)**

Jakob Dambon

**References**


---

**SVC_selection_control**  

**SVC Selection Parameters**

**Description**

Function to set up control parameters for `SVC_selection`. The underlying Gaussian Process-based SVC model is defined in `SVC_mle`. `SVC_selection` then jointly selects fixed and random effects of the GP-based SVC model using a penalized maximum likelihood estimation (PMLE). In this function, one can set the parameters for the PMLE and its optimization procedures (Dambon et al., 2021, <arXiv:2101.01932>).

**Usage**

```r
SVC_selection_control(
  IC.type = c("BIC", "cAIC_BW", "cAIC_VB"),
  method = c("grid", "MBO"),
  r.lambda = c(1e-10, 10),
  n.lambda = 10L,
  n.init = 10L,
  n.iter = 10L,
  CD.conv = list(N = 20L, delta = 1e-06, logLik = TRUE),
  hessian = FALSE,
  adaptive = FALSE,
  parallel = NULL,
  optim.args = list()
)
```

**Arguments**

- **IC.type** *(character(1))*
  Select Information Criterion.

- **method** *(character(1))*
  Select optimization method for lambdas, i.e., shrinkage parameters. Either model-based optimization (MBO, Bischl et al., 2017 <arXiv:1703.03373>) or over grid.

- **r.lambda** *(numeric(2))*
  Range of lambdas, i.e., shrinkage parameters.
n.lambda (numeric(1))
If grid method is selected, number of lambdas per side of grid.

n.init (numeric(1))
If MBO method is selected, number of initial values for surrogate model.

n.iter (numeric(1))
If MBO method is selected, number of iteration steps of surrogate models.

CD.conv (list(3))
List containing the convergence conditions, i.e., first entry is the maximum number of iterations, second value is the relative change necessary to stop iteration, third is logical to toggle if relative change in log likelihood (TRUE) or rather the parameters themselves (FALSE) is the criteria for convergence.

hessian (logical(1))
If TRUE, Hessian will be computed for final model.

adaptive (logical(1))
If TRUE, adaptive LASSO is executed, i.e., the shrinkage parameter is defined as \( \lambda_j := \lambda/|\theta_j| \).

parallel (list)
List with arguments for parallelization, see documentation of optimParallel.

optim.args (list)
List of further arguments of optimParallel, such as the lower bounds.

Value
A list of control parameters for SVC selection.

Author(s)
Jakob Dambon

References


Examples
# Initializing parameters and switching logLik to FALSE
selection_control <- SVC_selection_control(
   CD.conv = list(N = 20L, delta = 1e-06, logLik = FALSE)
)
# or
selection_control$CD.conv$logLik <- FALSE
Description

This package offers functions to estimate and predict Gaussian process-based spatially varying coefficient (SVC) models. Briefly described, one generalizes a linear regression equation such that the coefficients are no longer constant, but have the possibility to vary spatially. This is enabled by modeling the coefficients using Gaussian processes with (currently) either an exponential or spherical covariance function. The advantages of such SVC models are that they are usually quite easy to interpret, yet they offer a very high level of flexibility.

Estimation and Prediction

The ensemble of the function `SVC_mle` and the method `predict` estimates the defined SVC model and gives predictions of the SVC as well as the response for some pre-defined locations. This concept should be rather familiar as it is the same for the classical regression (`lm`) or local polynomial regression (`loess`), to name a couple. As the name suggests, we are using a maximum likelihood estimation (MLE) approach in order to estimate the model. The predictor is obtained by the empirical best linear unbiased predictor, to give location-specific predictions. A detailed tutorial with examples is given in a vignette; call `vignette("example",package = "varycoef")`. We also refer to the original article Dambon et al. (2021a) which lays the methodological foundation of this package.

With the before mentioned `SVC_mle` function one gets an object of class `SVC_mle`. And like the method `predict` for predictions, there are several more methods in order to diagnose the model, see `methods(class = "SVC_mle")`.

Variable Selection

As of version 0.3.0 of varycoef, a joint variable selection of both fixed and random effect of the Gaussian process-based SVC model is implemented. It uses a penalized maximum likelihood estimation (PMLE) which is implemented via a gradient descent. The estimation of the shrinkage parameter is available using a model-based optimization (MBO). Here, we use the framework by Bischl et al. (2017). The methodological foundation of the PMLE is described in Dambon et al. (2021b).

Author(s)

Jakob Dambon

References


**Examples**

```r
vignette("manual", package = "varycoef")
methods(class = "SVC_mle")
```
Index

* datasets
  house, 7
  AIC.SVC_mle (IC.SVC_mle), 8
  BIC.SVC_mle (IC.SVC_mle), 8
  coef.SVC_mle, 2
  cov_par, 3
  d.Lq, 3
  d.SCAD, 4
  fitted, 24
  fitted.SVC_mle, 5
  fullSVC_reggrid, 6
  house, 7, 7
  IC.SVC_mle, 8
  legend, 11, 12
  lm, 28
  loess, 28
  logLik.SVC_mle, 9
  Lq, 3, 9
  nearestdist, 24
  nlocs, 10
  nobs.SVC_mle, 11
  optim, 19, 20, 24
  optimParallel, 19, 23, 27
  plot.SVC_mle, 11
  predict.SVC_mle, 13, 20
  print.summary.SVC_mle, 15
  print.SVC_mle, 16
  residuals, 24
  residuals.SVC_mle, 17

SCAD, 4, 17
summary.SVC_mle, 16, 18
SVC_mle, 2, 3, 5, 8–14, 16–18, 19, 23–26, 28
SVC_mle_control, 3, 11, 20, 22, 25
SVC_selection, 25, 26
SVC_selection_control, 25, 26

varycoef, 28